Programação Visual
Trabalho de Laboratório nº 7

Objetivo	MVC – Introdução. Aplicações básicas com **WebAPI**.
Programa	Pretende-se continuar a implementação da aplicação **HospitEST** com o desenvolvimento de uma **API** que disponibilize as funcionalidades da aplicação (visualizar hospitais, médicos, pacientes…) através de serviços **REST**, para poderem ser utilizados por outras aplicações.
Regras	Implementar o código necessário e testar no fim de cada nível. Use as convenções de codificação adotadas para a linguagem C# e para o modelo MVC.

Descrição

Nível 1

- Faça a extração dos Materiais disponibilizados e execute o programa verificando que ele corre corretamente.
  Nota: Não esquecer de criar a migração inicial e atualizar a BD

- Adicione um controlador designado **`HospitalsApiController`** ao projeto. Para isso, faça **Add** > **Controller** e, dentro da janela, selecione o modelo **API Controller with actions, using Entity Framework**.

Corra o programa e verifique que consegue obter a lista de hospitais em formato json, usando os exemplos abaixo. Note que vai necessitar de modificar o valor do porto nestes exemplos:

- http://localhost:58279/api/HospitalsApi
  http://localhost:58279/api/HospitalsApi/d5895af1-0207-4080-b933-67746e856de3

  Nota: *verifique o valor do porto no ficheiro* ***launchSettings.json*** *na pasta das propriedades ou no browser quando arranca com a aplicação.*
- Para verificar os restantes métodos de CRUD vai-se utilizar o Swagger. Sendo assim comece por instalar o package necessário através do seguinte comando:
  ```
  install-package Swashbuckle.AspNetCore
  ```
- Agora no ficheiro **Program.cs** instale o serviço do Swagger:
  ```
  builder.Services.AddEndpointsApiExplorer();
  builder.Services.AddSwaggerGen();
  ```
- No mesmo ficheiro depois de `var app = bulider.Build();` coloque o seguinte código:
  ```
  app.UseSwaggerUI();
  app.UseSwagger();
  ```
- Usando o swagger em http://localhost:58279/swagger/index.html, testes os restantes métodos da API.

Programação Visual
Trabalho de Laboratório nº 7

Nível 2

- Vamos agora desenvolver uma aplicação cliente para a **WebApi** criada no nível anterior. Aplicações clientes típicas podem ser aplicações Web, Apps de Smartphones Android/IPhone, Windows Desktop, etc. Para efeitos ilustrativos, é suficiente utilizar um menu de opções feito numa janela de consola, que sirva para testar a interação com a aplicação **HospitEST.**
- Sendo assim, **adicione à solução** um novo projeto do tipo **Console App (.Net Core),** com o nome **HospitESTClient**. Para facilitar o desenvolvimento deste novo projeto, defina-o como principal (**Set as Startup Project**).
- Na janela do projeto, adicione a referência ao projeto **HospitEST** (add project reference).
- Selecione o projeto **HospitESTClient** na consola do **NuGet** e instale o seguinte pacote:

  ```
  Install-Package Microsoft.AspNet.WebApi.Client
  ```
- Adicione ao projeto **HospitESTClient**, a classe `HospitESTClientUtils` fornecida com este enunciado.
- Abra agora o ficheiro fornecido com o nome **Cliente_WebApi_ilustrativo.txt** e copie o seu conteúdo para dentro da classe `Program` do projeto **HospitESTClient**.
- No novo conteúdo da classe `Program`, Corrija a porta do URLque está guardado no ficheiro **launchSettings.json** do projeto **HospitEST.**
- Compile e verifique a ausência de erros.

Nível 3

- Neste passo, vai necessitar de ter a funcionar simultaneamente os projetos **HospitEST** e **HospitESTClient**. No **Solution Explorer**, selecione o projeto **HospitESTClient** e execute a opção "**Debug > Start New Instance**".
- De forma similar, execute o projeto **HospitEST**.
- Experimente a opção 1 do menu fornecida já implementada. Constate o seu funcionamento correto.
  Nota: ambos os projetos precisam de correr em simultâneo, se ocorrer um erro de ligação rejeitada, verifique que ambos estão a correr corretamente.

- Implemente agora o método `GetHospitalAsync` de forma que possa visualizar um hospital fornecendo o seu Id.
- Teste o funcionamento da API.

Programação Visual
Trabalho de Laboratório nº 7

Nível 4

- Implemente agora os métodos **`CreateHospitalAsync`**, **`UpdateHospitalAsync`** e **`DeleteHospitalAsync`**.
- Chame cada um dos métodos implementados a partir da opção do menu existente no método **`Main`.** Constate o bom funcionamento dos métodos referidos.

Nível 5

- Acrescente agora uma opção de menu "`[ 6 ] Consultar Médicos de um Hospital`" que a partir do Id do hospital liste os médicos desse hospital. Deve ser mostrado o nome do Hospital e listar os seus médicos mostrando o nome do médico seguida da área entre parênteses.

Desafio

- Termine a construção da API adicionando os CRUDs para os médicos e pacientes.
- Atualize o menu para 3 secções (para cada modelo), a primeira opção fornecendo o menu já familiarizado nos níveis anteriores, a segunda opção fornecendo uma lista de comandos fazendo o CRUD dos médicos, e a terceira opção fornecendo o mesmo mas para os pacientes.

Notas

Para os identificadores siga as convenções adotadas pelo C#, nomeadamente:

- A notação camelCase para o nome das variáveis locais e identificadores privados.
- A notação PascalCase para os nomes públicos dos métodos e classes
- Não utilize o símbolo de qualquer forma '_' nos identificadores
- Não use abreviaturas